MASSACHUSETTS INSTITUTE OF TECHNOLOGY

ARTIFICIAL INTELLIGENCE LABORATORY

Memo No. 328 March 1975

HEURISTIC TECHNIQUES IN COMPUTER AIDED CIRCUIT ANALYSIS

by

Gerald Jay Sussman and Richard Matthew Stallman

Abstract:

We present EL, a new kind of circuit analysis program. Whereas other circuit analysis systems rely on classical, formal analysis techniques, EL employs heuristic "inspection" methods to solve rather complex DC bias circuits. These techniques also give EL the ability to explain any result in terms of its own qualitative reasoning processes. EL's reasoning is based on the concept of a "local one-step deduction" augmented by various "teleological" principles and by the concept of a "macro-element". We present several annotated examples of EL in operation and an explanation of how it works. We also show how EL can be extended in several directions, including sinusoidal steady state analysis. Finally, we touch on possible implications for engineering education. We feel that EL is significant not only as a novel approach to circuit analysis but also as an application of Artificial Intelligence techniques to a new and interesting domain.

Work reported herein was conducted at the Artificial Intelligence Laboratory, a Massachusetts Institute of Technology research program supported in part by the Advanced Research Projects Agency of the Department of Defence and monitored by the Office of Naval Research under contract number N0014-70-A-0362-0005.

Introduction:

Every engineering student is taught "formal" methods for analyzing electrical networks. The practicing engineer quickly discovers that he hardly ever uses those methods. He finds himself solving most problems by "inspection" techniques. These inspection methods are said to be "informal" and to spring from the mysterious land of "experience". In an effort to formalize these intuitive notions we have written EL, a new kind of electrical network analysis program [1]. The literature is replete with powerful and useful circuit analysis systems [2] which implement the formal methods. What is novel about this program is its heuristic approach to network analysis and its consequent <u>ability to explain the basis of its deductions</u>. Although the current version of EL can only handle DC bias analysis, with a rather crude transistor model at that, the approach leads quickly to rather impressive results.

The best way to understand EL is to follow it out on some examples. We first present several example conversations with EL. We have annotated the examples to extract the reasoning used. Then we give a general discussion of the techniques employed to implement the results illustrated. Finally, we discuss the possible extensions and ultimate limitations of these techniques. We also touch on possible implications for education. In the text that follows the Gothic font will be used to indicate interactions with the computer -- lower case is typed by the user, upper case by the computer. The commentary is in the Roman font.

Example 1:

Let us first consider a simple four transistor amplifier circuit [3] with no multistage feedback (See Figure 1.). We see that this is a direct-coupled amplifier. Its input

is a common-emitter Darlington pair, followed by a simple common-emitter amplifier, followed by an emitter-follower output circuit. We encode this wiring diagram for EL as follows:

```
(wire 4t-amp
 (node (vcc 15) (gnd 0) input output)
 (transistor (q1 .6) (q2 .6) (q3 .6) (q4 .6))
 (resistor (r1 370000) (r2 100000) (r3 10000) (r4 1800)
           (r5 5600) (r6 3900) (r7 6800))
 (connect
  (vcc (#1 r1) (#1 r3) (#1 r5) (c q4))
  (gnd (#2 r2) (#2 r4) (#2 r6) (#2 r7))
  (input (#2 r1) (#1 r2) (b q1))
  ((#2 r3) (c q1) (c q2) (b q3))        ;NODE1
  ((e q1) (b q2))                       ;NODE2
  ((e q2) (#1 r4))                      ;NODE3
  ((#2 r5) (c q3) (b q4))               ;NODE4
  ((e q3) (#1 r6))                      ;NODE5
  (output (e q4) (#1 r7)))
 (hint (vd r1 r2)))

DONE
```

I hope that this method of specifying the wiring diagram is pretty clear. The diagram gets a name, 4T-AMP. Next, the named nodes are declared; in this case there are only four of them -- VCC, GND, INPUT, and OUTPUT. There are other nodes in the network but they are not given names by the user. EL chooses names for them. We have indicated EL's names by "comments" beginning with semicolons. Notice that two of the named nodes are given default fixed voltages; the others are left with unknown voltages. Four silicon, NPN transistors are declared -- they have a default base-emitter voltage of .6 volt. PNP transistors would have negative base-emitter voltages, and germanium transistors would have voltage drops of magnitude .3. Seven resistors are similarly declared with their resistance values in ohms. Then, all connections are made. Each component has certain

terminals -- a resistor has a #1 and a #2, a transistor has a B, E, and C. Each expression in the CONNECT clause is the set of terminals which are identified to make up a single node. This section is used to assign the previously declared names of the named nodes to particular connections. Finally, the encoded wiring diagram contains a section for the declaration of certain hints which are necessary for the proper operation of the analysis program. In this case we see that R1 and R2 are declared to together constitute a voltage divider. The reason for the necessity of such <u>teleological</u> hints will be discussed later [4].

After the wiring diagram is specified the we ask EL to perform the analysis:

```
(analyze '4t-amp)

(Q1 CURRENT E 0.0)
(INPUT VOLTAGE V7)
(R1 CURRENT ME (&+ 4.0540541E-5 (&* -2.7027027E-6 V7)))
(R2 CURRENT #1 (&+ 4.0540541E-5 (&* -2.7027027E-6 V7)))
(V7 VARVAL 3.19148964)
(NODE2 VOLTAGE 2.59148955)
(NODE3 VOLTAGE 1.99148953)
(R4 CURRENT ME 1.10638307E-3)
(Q2 CURRENT E -1.10638307E-3)
(R3 CURRENT #2 -1.10638304E-3)
(NODE1 VOLTAGE 3.9361696)
(NODE5 VOLTAGE 3.3361697)
(R6 CURRENT ME 8.55428135E-4)
(Q3 CURRENT E -8.55428135E-4)
(R5 CURRENT #2 -8.5542817E-4)
(NODE4 VOLTAGE 10.2096023)
(OUTPUT VOLTAGE 9.609602)
(R7 CURRENT ME 1.41317676E-3)
(Q4 CURRENT E -1.41317676E-3)
DONE
```

Now let's examine how the analysis proceeds. First, EL <u>notices</u> that the current into the emitter of Q1 is 0.0 amperes. Of course, this is false; but it is very nearly true -- it is based

on the assumption that the base current of a transistor is likely to be insignificant. This is not an assumption about the intrinsic properties of the transistor but rather on its extrinsic properties -- how people use transistors in design. Such teleological assumptions are the source of much of the power, and most of the limitations of EL. Next, EL concentrates on the input voltage divider. An abstract voltage unknown, V7, is assumed at the center of the divider. This makes it possible to compute an expression for the current through R1. Since no current goes into the base of transistor QI (ha, ha!) EL deduces that all of the current coming out of R1 goes into R2. But then, Ohm's law is applied at R2 to get the voltage on the INPUT node. The resulting expression is set equal to V7. This equation has only one unknown -- it is easy to solve -- thus EL finds a value for V7. Now, NODE2 is the second anonymous node in the wiring diagram -- where the emitter of QI connects to the base of Q2. EL makes the assumption that transistors are run in their active region unless otherwise hinted -- another teleological assumption. Thus, the base-emitter drop of QI is assumed to be the nominal .6 volts so NODE2 is assigned a voltage .6 volts less than V7. NODE3 is at the emitter of Q2. It is assigned a voltage in a similar manner. This makes it possible to calculate the current through R4. This current must all go through the transistor, Q2, and come out of its collector. It then finds its way into R3 (since none goes into the base of Q3 and EL has already deduced that QI's current is 0.0). This allows EL to deduce the voltage at node NODE1 (Q2's collector).

We could continue to explain these deductions, ad nauseum. We would see that they all -- except for the assumption of an unknown voltage at the center of the voltage divider -- have an essentially local flavor [5]. Each network element or node is a little process attempting to assign any network unknown that can be deduced locally from

knowns -- but more about the implementation later. EL is very conscious about how it makes its deductions. As it makes them it takes notes on the process. Thus we can ask it to justify the claim that it knows, say, the value of the current through R3:

```
(why r3 current)

((R3 CURRENT) <= ((Q3 CURRENT B) (Q2 CURRENT C) (Q1 CURRENT C)) - KCL)
((Q3 CURRENT B) GIVEN)
((Q2 CURRENT C) <= ((R4 CURRENT #1)) - KCL)
((Q1 CURRENT C) <= ((Q2 CURRENT B)) - KCL)
((R4 CURRENT #1) <= ((NODE3 VOLTAGE) (GND VOLTAGE) (R4 RESISTANCE)) - OHM)
((Q2 CURRENT B) GIVEN)
((NODE3 VOLTAGE) <= ((Q2 VBE) (NODE2 VOLTAGE)) - BE)
((R4 RESISTANCE) GIVEN)
((Q2 VBE) GIVEN)
((NODE2 VOLTAGE) <= ((Q1 VBE) (INPUT VOLTAGE) (V7 VARVAL)) - BE)
((Q1 VBE) GIVEN)
((V7 VARVAL) <= ((INPUT VOLTAGE) (GND VOLTAGE)
                 (R2 CURRENT #1) (R2 RESISTANCE)) - OHM)
((R2 CURRENT #1) <= ((Q1 CURRENT B) (R1 CURRENT #2)) - KCL)
((Q1 CURRENT B) GIVEN)
((R1 CURRENT #2) <= ((VCC VOLTAGE) (INPUT VOLTAGE) (R1 RESISTANCE)) - OHM)
((INPUT VOLTAGE) <= ((R1 RESISTANCE) (R2 RESISTANCE)
                     (VCC VOLTAGE) (GND VOLTAGE)) - VD-DEMON)
((R1 RESISTANCE) GIVEN)
((VCC VOLTAGE) GIVEN)
((R2 RESISTANCE) GIVEN)
((GND VOLTAGE) GIVEN)
        QED
```

Here we see the "proof" that the current indicated can be deduced from elementary assumptions. Each line of the proof is the justification of the deduction of the first quantity named in that line. The list after the arrow is the quantities on which the deduction was based. The last element of the line is the "law" by which the deduction was made. Besides such familiar laws as KVL, KCL, and OHM, there are some other, less familiar ones as BE, governing base-emitter voltages of transistors, and VD-DEMON, which mediates the hint about voltage-dividers. Note that the line beginning "INPUT

VOLTAGE" represents the decision to introduce a symbolic variable ("V7", in this instance) to represent the voltage at the node named INPUT. The reasons given are the facts that show that the node INPUT is in the middle of a voltage divider. The line beginning "V7 VARVAL" represents the actual determination of the value of V7, with the bases being all the quantities entering into the equation which was used to solve for V7.

EL remembers not only how it arrived at each fact it knows, but also all the ways each fact was used in deductions. Thus, we can ask EL to forget the assumed value of one of the initially given quantities, and also all conclusions depending on that assumption. Here, we tell EL to try a new value for the resistance of R6:

```
(change r6 resistance 3200.0)

(R6 RESISTANCE 3200.0)
(R6 CURRENT ME 1.04255303E-3)
(Q3 CURRENT E -1.04255303E-3)
(R5 CURRENT #2 -1.042553E-3)
(NODE4 VOLTAGE 9.161703)
(OUTPUT VOLTAGE 8.56170272)
(R7 CURRENT ME 1.25907393E-3)
(Q4 CURRENT E -1.25907393E-3)
DONE
```

EL first forgot the initially assumed value of R6's resistance and all deduced quantities depending on it. Then, EL made the new assumption about the resistance of R6, and proceded to make all possible deductions from that assumption, along with those previous conclusions that it had not been necessary to forget, using the same mechanism as before. All of the forgotten quantities were rededuced, but with appropriately different values.

Example 2:

This circuit (See Figure 2.) describes another direct-coupled amplifier [6]. Here we see two new sources of complexity. There is a multistage feedback loop and a complementary pair. We introduce this example because the previous example might give you the false impression that EL's local deductive scheme gives rise to an overall reasoning which is causal in nature. Causal reasoning often causes difficulty in analysis of systems with feedback because of instabilities in relaxation. The kind of reasoning done by EL, however, gets the answer without relaxation.

Let us first understand how this circuit is supposed to work -- we look at the real causality. In this circuit Q107 and Q108 form a complementary pair which is supposed to hold the node labeled OUTPUT at a voltage just somewhat higher than the voltage on the node labeled INPUT. If the output voltage is higher than it ought to be, Q105 conducts more than it ought to. This means more current enters the base of Q106, causing it to conduct more, pulling down the bases of Q107 and Q108, and thus restoring the output node to its correct voltage. This is the negative feedback path. Now let's see what EL does with this circuit -- first we must give it the wiring diagram:

```
(wire gr-58
 (node (vcc 25.0) (gnd 0) input output)
 (resistor (r130 180000) (r131 120000) (r132 2700)
           (r135 4700) (r136 150) (r137 1800)
           (r138 2.7) (r139 2.7))
 (transistor (q105 -.6) (q106 .6) (q107 .6) (q108 -.6))
 (connect
  (vcc (#1 r130) (#1 r137) (c q107))
  (input (#2 r130) (b q105) (#1 r131))
  ((#1 r132) (c q105) (b q106))                    ;NODE1
  ((e q105) (#2 r135))                             ;NODE2
  (gnd (#2 r131) (#2 r132) (e q106) (c q108))
  ((c q106) (b q108) (#2 r136))                    ;NODE3
  ((#1 r136) (b q107) (#2 r137))                   ;NODE4
  ((e q107) (#1 r138))                             ;NODE5
  (output (#2 r138) (#1 r135) (#1 r139))
  ((#2 r139) (e q108)))                            ;NODE6
 (hint (vd r130 r131) (cp q107 q108)) )

DONE
```

One new type of hint appears here: (CP Q107 Q108). It tells EL to regard Q107 and Q108 as a complementary pair -- more about this later. Let's now examine the analysis of this circuit:

```
(analyze 'gr-58)

(NODE1 VOLTAGE 0.60000002)
(R132 CURRENT ME 2.2222223E-4)
(Q105 CURRENT C -2.22222224E-4)
(R135 CURRENT #2 -2.22222224E-4)
(INPUT VOLTAGE V9)
(R130 CURRENT ME (&+ 1.38888892E-4 (&* -5.5555556E-6 V9)))
(R131 CURRENT #1 (&+ 1.38888892E-4 (&* -5.5555556E-6 V9)))
(V9 VARVAL 10.000001)
(NODE2 VOLTAGE 10.6000013)
(OUTPUT VOLTAGE 11.6444454)
```

Let me interrupt the output stream here to insert an observation: EL has determined the voltage on the output node without making any deductions about the properties of

transistors QI07 or QI08 -- but it is these transistors which actually <u>cause</u> the output node to have that particular voltage! It seems that the teleological assumptions about QI05 and QI06 -- the assumptions that the designer intended that they run in their active regions -- is sufficient to "force" the voltage on the output node to be determined. With this information in hand EL goes on to deduce other properties of the circuit — in particular, the approximate voltages on the bases of QI07 and QI08:

```
(NODE4 VOLTAGE (&+ 11.6444454 V10))
(R137 CURRENT ME (&+ 7.4197524E-3 (&* -5.5555555E-4 V10)))
(R136 CURRENT #1 (&+ 7.4197524E-3 (&* -5.5555555E-4 V10)))
(QI06 CURRENT C (&+ 7.4197524E-3 (&* -5.5555555E-4 V10)))
(NODE3 VOLTAGE (&+ 10.5314827 (&* 1.08333337 V10)))
(V10 VARVAL 0.53422212)

DONE
```

Notice how, as soon as the voltage at the node OUTPUT has been determined, EL assumes values for the voltages at the bases of the transistors involved: at NODE3 the voltage assumed is 11.64+V10, while that at NODE4, the other transistor's base, it is 11.64-V10. The complementary pair declaration created a demon which embodies the assumption that the voltages on the transistor bases are symmetrical around the midpoint of the pair -- in this case, the node OUTPUT. If one of those three voltages receives a value, the other two will be given values in terms of it and a symbolic variable made for the occasion. Thus, if one base voltage were given the value 10, the OUTPUT voltage would receive the value 10-X; the other base, the value 10-2X. The value of X could be determined if the two unknown voltages were related in any other way. In this case the network of resistors around NODE3 and NODE4, leads to an independent determination of the voltage at NODE4

(10.53+1.063*V10) which, together with the complemetary pair assumption of 11.64-V10, allows V10 to be determined. The complementary-pair hint also tells EL that the transistors may be cut off, so their base-emitter voltages should not be assumed to be the usual .6 or .3 volts. As in the previous example, we can ask EL how it arrived at any of its conclusions:

```
(why output voltage)

((OUTPUT VOLTAGE) <= ((NODE2 VOLTAGE) (R135 CURRENT #2)
                      (R135 RESISTANCE)) - OHM)
((NODE2 VOLTAGE) <= ((Q105 VBE) (INPUT VOLTAGE) (V9 VARVAL)) - BE)
((R135 CURRENT #2) <= ((Q105 CURRENT E)) - KCL)
((R135 RESISTANCE) GIVEN)
((Q105 VBE) GIVEN)
((V9 VARVAL) <= ((INPUT VOLTAGE) (GND VOLTAGE)
                 (R131 CURRENT #1) (R131 RESISTANCE)) - OHM)
((Q105 CURRENT E) <= ((Q106 CURRENT B) (R132 CURRENT #1)) - KCL)
((R131 CURRENT #1) <= ((Q105 CURRENT B) (R130 CURRENT #2)) - KCL)
((Q106 CURRENT B) GIVEN)
((R132 CURRENT #1) <= ((NODE1 VOLTAGE) (GND VOLTAGE)
                       (R132 RESISTANCE)) - OHM)
((Q105 CURRENT B) GIVEN)
((R130 CURRENT #2) <= ((VCC VOLTAGE) (INPUT VOLTAGE)
                       (R130 RESISTANCE)) - OHM)
((INPUT VOLTAGE) <= ((R130 RESISTANCE) (R131 RESISTANCE)
                     (VCC VOLTAGE) (GND VOLTAGE)) - VD-DEMON)
((R130 RESISTANCE) GIVEN)
((VCC VOLTAGE) GIVEN)
((R131 RESISTANCE) GIVEN)
((NODE1 VOLTAGE) <= ((Q106 VBE) (GND VOLTAGE)) - BE)
((GND VOLTAGE) GIVEN)
((R132 RESISTANCE) GIVEN)
((Q106 VBE) GIVEN)
        QED
```

Example 3:

In this example (See Figure 3.) we show how the local one-step deductions of EL handle a classical problem of network analysis -- the ladder network. This network is the

basis of many important filter networks -- we will come back to this again when we talk about frequency domain analysis. As usual, we must first give EL the wiring diagram:

```
(wire ladder
  (node (gnd 0.0) (input 1.0) output)
  (resistor (r1 1) (r2 2) (r3 1) (r4 2) (r5 1) (r6 1))
  (connect
      (gnd (#2 r2) (#2 r4) (#2 r6))
      (input (#1 r1))
      ((#2 r1) (#1 r2) (#1 r3))        ;NODE1
      ((#2 r3) (#1 r4) (#1 r5))        ;NODE2
      (output (#2 r5) (#1 r6))))
DONE
```

We next ask EL to analyze the network:

```
(analyze 'ladder)
DONE
```

Nothing happened! There are no immediate one-step deductions that can be made. No resistor has three out of four of its parameters defined and no node has all except one of its branch currents defined. Does this mean that EL finds the problem hopeless? We have never told EL just what we were after -- let's ask for the voltage on the node labelled OUTPUT:

```
(what output voltage)

(OUTPUT VOLTAGE V3)
(R6 CURRENT ME V3)
(R5 CURRENT #2 (&* -1.0 V3))
(NODE2 VOLTAGE (&* 2.0 V3))
(R4 CURRENT ME V3)
(R3 CURRENT #2 (&* -2.0 V3))
(NODE1 VOLTAGE (&* 4.0 V3))
(R1 CURRENT ME (&+ 1.0 (&* -4.0 V3)))
(R2 CURRENT #1 (&+ 1.0 (&* -6.0 V3)))
(V3 VARVAL 0.125)

0.125
```

We seem to have gotten the answer -- the output voltage is 1/8 volt. EL used an old trick [7] which might be called "wishful thinking". EL looked for the answer and determined that it was unknown. It then assumed that it knew the answer -- it postulated a formal variable for the answer. The consequences of this assumption were then worked out. In particular, if the output voltage is known then we get a value for the current through R6. This same current goes through R5, giving us the voltage at the other side of R5 (NODE2). We can now get the current through R4 since we know the voltage at each of its terminals. KCL now gives us the current through R3. We use the current through R3 to get the voltage at the top of R2 (NODE1). This makes it possible to deduce the current through R1 by Ohm's law. KCL is then applied at NODE1 getting the current through R2. Happily, the voltage is known at both sides of R2. This application of Ohm's law as a consistency check results in an equation in one unknown -- the original assumed voltage -- to solve.

How EL works:

Now that we see what EL can do it is time to examine how it works. We feel that

the ideas behind the implementation elucidate various fundamental problem solving notions. By now you probably realize that one essential ingredient in EL is the idea of a "local one-step deduction". Inside of EL each conceptual object in the electrical network under consideration (eg. a resistor, transistor, or node) is modelled as a data structure with various "slots" which can be "filled" with data. Each of these structures also has a TYPE which specifies what "laws" apply to that structure. A law, Ohm's law for example, is a set of procedures relating the resistor's slots for its resistance, the voltages at the two nodes attached to its terminals, and its branch current, so that if all except one of the slots is filled the last can be filled. Each instance of resistor, node, or other object which is part of a circuit wiring diagram knows what other parts it is connected to and thus, what other parts would be interested if one of its slots becomes filled by the application of a law. When such an event takes place, the relevent other parts are awakened so that their laws may be applied to the situation. Thus, a new piece of information may be locally deduced in one place in the network but consequences of this deduction may propagate from element to element all over the network. Because of the strong connectedness of electrical networks it is often the case that an element is awakened by the filling of its last slot by a neighbor. In this situation a law may not be used to deduce anything new but it is applied to check the consistency of the assumptions which have been made. Such a consistency check may, however, result in a deduction -- ie. the assignment of a value to a formal unknown (as we shall see).

Of course, local one-step deductions are not sufficient to solve most, or even many, networks of interest. It is sometimes necessary to take a somewhat more global view. Consider, for example, the situation of two resistors in series, as in a voltage divider.

Although the voltage may be known at both ends of the divider there is no one-step deduction which will get the answer at the center. The problem is that neither resistor has enough information to fill a slot. Each resistor has one terminal voltage and its own resistance filled but both the current through it and the voltage at the midpoint node are still unknown. The essence of why the situation is solvable, however, is the more global assertions that "Whatever current goes through one resistor goes through the other." and "The voltage at the top of the bottom resistor is the same as the voltage at the bottom of the top resistor." -- the simultaneous constraints. Just how can we handle this kind of situation? The answer is hidden in our description of the situation! We described the situation in terms of the concept of a voltage divider. A voltage divider is a composite element with three terminals made up of simpler elements working together to achieve the purpose of producing a particular voltage at its midpoint. The hint declaration compiles into a demon -- actually a macro-element -- whose law senses the voltage slots of the top and bottom nodes of the voltage divider. If they are filled it assigns a newly generated symbol of TYPE VARIABLE to the voltage slot of the midpoint of the divider. One resistor is awakened, assigning the current through it in terms of this abstract potential. KCL at the midpoint then assigns this current to the other resistor which then awakens to run a consistency check. In testing the equality EL discovers that there is an unassigned variable in the equation under test. The equation is then solved and the variable assigned. The voltage divider demon uses the abstract variable manipulation system rather than assigning the voltage at the midpoint itself (from the voltage divider formula, which it could know) so that the voltage can be correctly determined even if there is a significant current withdrawn from the midpoint. The concept of a macro-element is independent of the concept of an abstract formal unknown.

Is this just an ad hoc method? -- or is there something more fundamental here worthy of consideration? We believe the latter, the concept of a macro-element with a teleology is essential to solving hard problems if we are not to get bogged down in details. In support of this idea is the fact that human circuit designers constantly use such macro components as op-amps, and-gates and flip-flops. The complementary pair macro of Example 2 is a good example where we see that the characteristics of the compound element are not trivially deducible from the elementary characteristics of its parts. The hint demon mechanism is a start at building a hierarchy of higher order plan fragments and other teleological commentary to direct the process of hypothesis formation in reasoning about such causal systems. It is one method of transforming a global deductive process into a local one which can be handled by one-step deductions in an efficient way.

Extensions:

We see several directions in which EL can be extended. We can give it more knowledge of electricity and we can try to give it even more powerful problem-solving capabilities. As it stands, EL knows nothing about signals, inductors or capacitors -- only DC bias analysis. Thus, a logical place to begin to discuss extensions is to examine what it would take to make EL capable of incremental analysis.

Let us first consider the circuit of Figure 4. This is a standard capacitance-coupled common-emitter amplifier. Just what kinds of new features would EL need to be able to give an account of the signal as well as the DC bias? Suppose that we augment EL as follows: For each current or voltage slot in the old program let there be a pair of slots --

(4) stop Z-motion.

The program for this procedure is:

```
(DEFUN DROP-INTO-L ( )
       (DX= shift) (WAIT ' (?X))
       (FZ= small-contact-force)(WAIT '(?FZ))
       (FX= small-sliding-force)
       (WAIT '(OR (?FX) (SEQ (GETM FZ) 0 Threshold)))
       (DZ= 0.0)   )
```

Note; (WAIT '(?FX)) waits until the SHAFT contacts the left wall of the hole. (WAIT '(SEQ (GETM FZ) 0 Threshold)) checks the occurrence of the "drop". Either event will complete the desired action.

(b) MATE: When DROP-INTO has been completed, SHAFT should be at least partially in the hole of SPACER2. So, the following procedure can guide the position of SHAFT exactly to the center of SPACER2.

(1) move SHAFT toward "+Y" until it finds an edge of SPACER2

and get the Y-position of SHAFT ==> Y1

(2) move SHAFT toward "-Y" until it finds another edge of SPACER2

and get the Y-position of SHAFT ==> Y2

(3) move SHAFT to (Y1 + Y2)/ 2

this motion locates the SHAFT in the actual center along the Y axis.

similarly;

(4) find "+X" edge ==> X1

(5) find "-X" edge ==> X2

(6) move SHAFT to (X1 + X2)/ 2

What follows is a program that performs this procedure.

```
(DEFUN MATE-Y ( )
      (PROG (X1 Y1)
        (FY= small-edge-finding-force-to-"+Y") (WAIT '(?FY))
        (SETQ Y1 (GETMF YPOS))
        (FY= small-edge-finding-force-to-"-Y") (WAIT '(?FY))
        (Y= (//$ (+$ Y1 (GETMF YPOS)) 2.0)) (WAIT '(?Y))
        (FX= small-edge-finding-force-to-"+X") (WAIT '(?FX))
        (SETQ X1 (GETMF XPOS))
        (FX= small-edge-finding-force-to-"-X") (WAIT '(?FX))
        (X=  (//$ (+$ X1 (GETMF XPOS)) 2.0)) (WAIT '(?X)) ))
```

(c) PUSH-INTO-Y: SPACER2 is moved down along the SHAFT until it arrives at the end. During this process, the X and Y position of the SHAFT should accommodate to that of SPACER2. This is easily done by controlling X and Y forces to be zero. The program for this is:

```
(DEFUN PUSH-INTO-Y ( )
        (FX= 0) (FY= 0) (FZ= inserting-force) (WAIT '(?FZ)) )
```

The sequence of DROP-INTO-L, MATE-Y, and PUSH-INTO-Y gives a simple stereotyped action for the peg-into-hole assembly with loose fit.

```
(DEFUN PEG-HOLE-L ( )
        (DROP-INTO-L) (MATE-Y) (PUSH-INTO-Y) )
```

Actually, we can omit the function MATE-Y, because during the following function PUSH-INTO-Y, the X, Y position of the SHAFT adapts to the center of SPACER2. Thus we have an even simpler stereotyped action for this task.

```
(DEFUN PEG-HOLE-L2 ( ) (DROP-INTO-L) (PUSH-INTO-Y) )
```

3.1.2 Close Fit.

The other class of fit is "close fit", defined by ($c < \sqrt{2} * e$). The BEARING-CYLINDER pair, shown in figure 7, is a good example of this. The function DROP-INTO-L will not reliably perform the drop-into process of a pair with close fit, because the tolerance circle does not cover

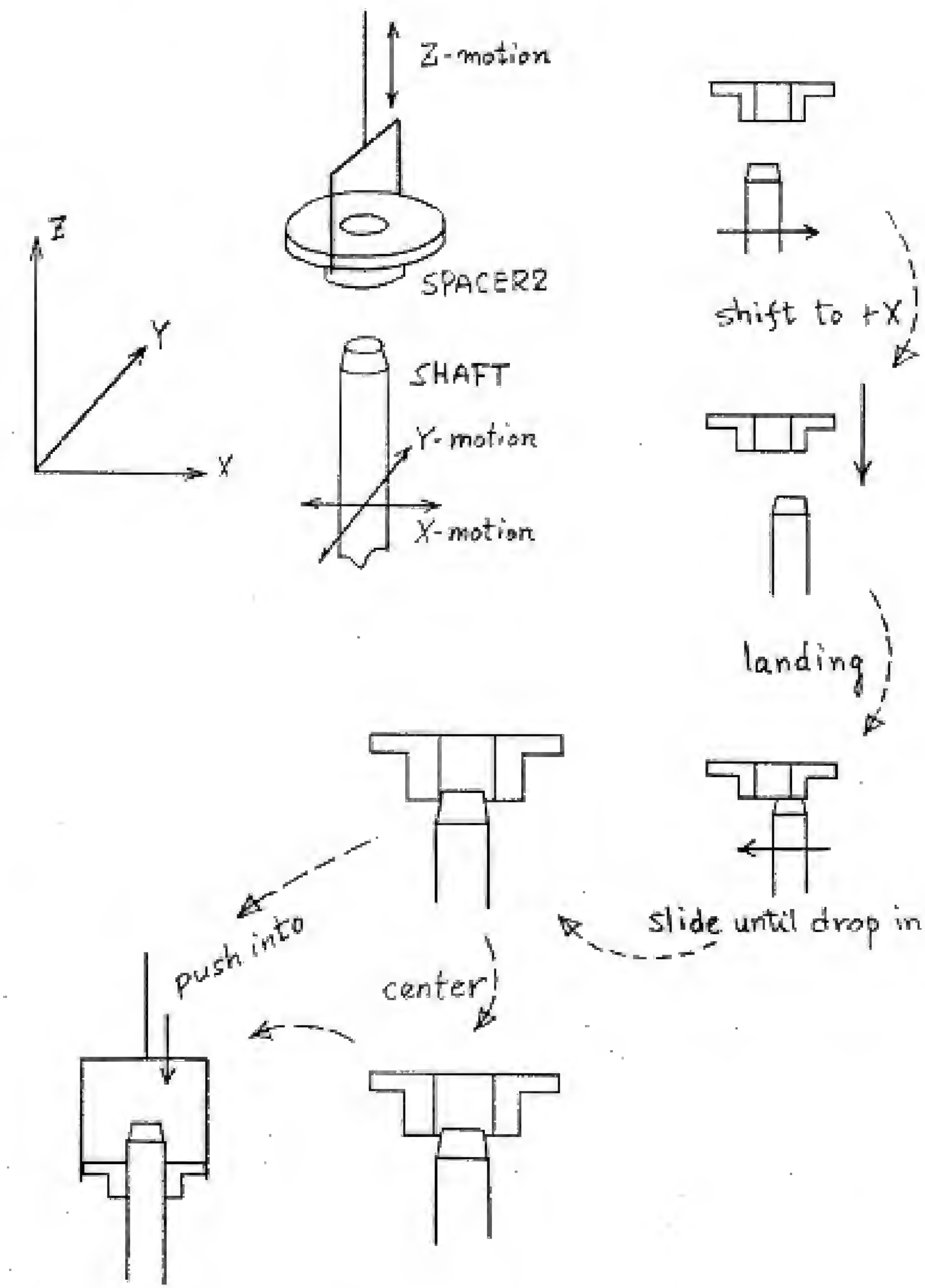


Figure 6. Peg-into-Hole assembly with "loose fit".

all of the error square. The simplest technique for solving this difficulty is that of tilting the peg or the hole. By tilting, the tolerance circle is equivalently enlarged up to the size of hole, which is much larger than the error square. For example, the diameter of the tolerance circle of the BEARING—CYLINDER pair is 0.001 inch without tilting, and 0.688 inch with tilting. Note that $\sqrt{2} * e$ of the system is about 0.003 inch. So, if we apply the tilting technique to the pair with close fit, we can treat it as if it were a loose fit. The following is a program to do the drop-into process for a pair with close fit.

```
(DEFUN DROP-INTO-C ( )
       (DR= 0.1) (DY= shift) (WAIT '(AND (?R) (?Y))
       (FX= landing-force) (WAIT '(?FX))
       (DX= 0.0) )
```

After the DROP-INTO-C is completed, CYLINDER must be aligned to the orientation of SHAFT and Y, Z position of BEARING must be adapted to CYLINDER. The procedure is:

(1) move BEARINGs toward "+Z" until it finds an edge of CYLINDER, and get the Z-position ==> Z1

(3) move BEARINGs toward "-Z" until it finds another edge of CYLINDER, and get the Z-position ==> Z2

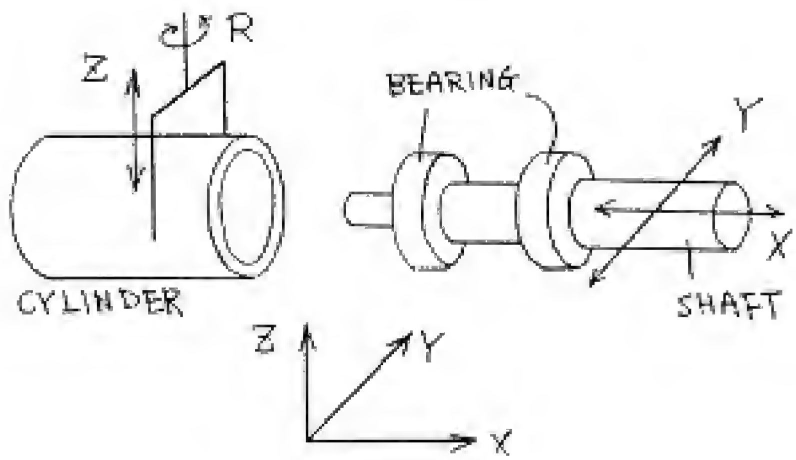
(3) move to (Z1 + Z2) / 2

(4) move BEARINGs to +Y direction until it contacts the CYLINDER

(5) keep above contact and apply small force in +X direction

(6) null the Z-force and rotate CYLINDER to R = 0.0

The program for this is:



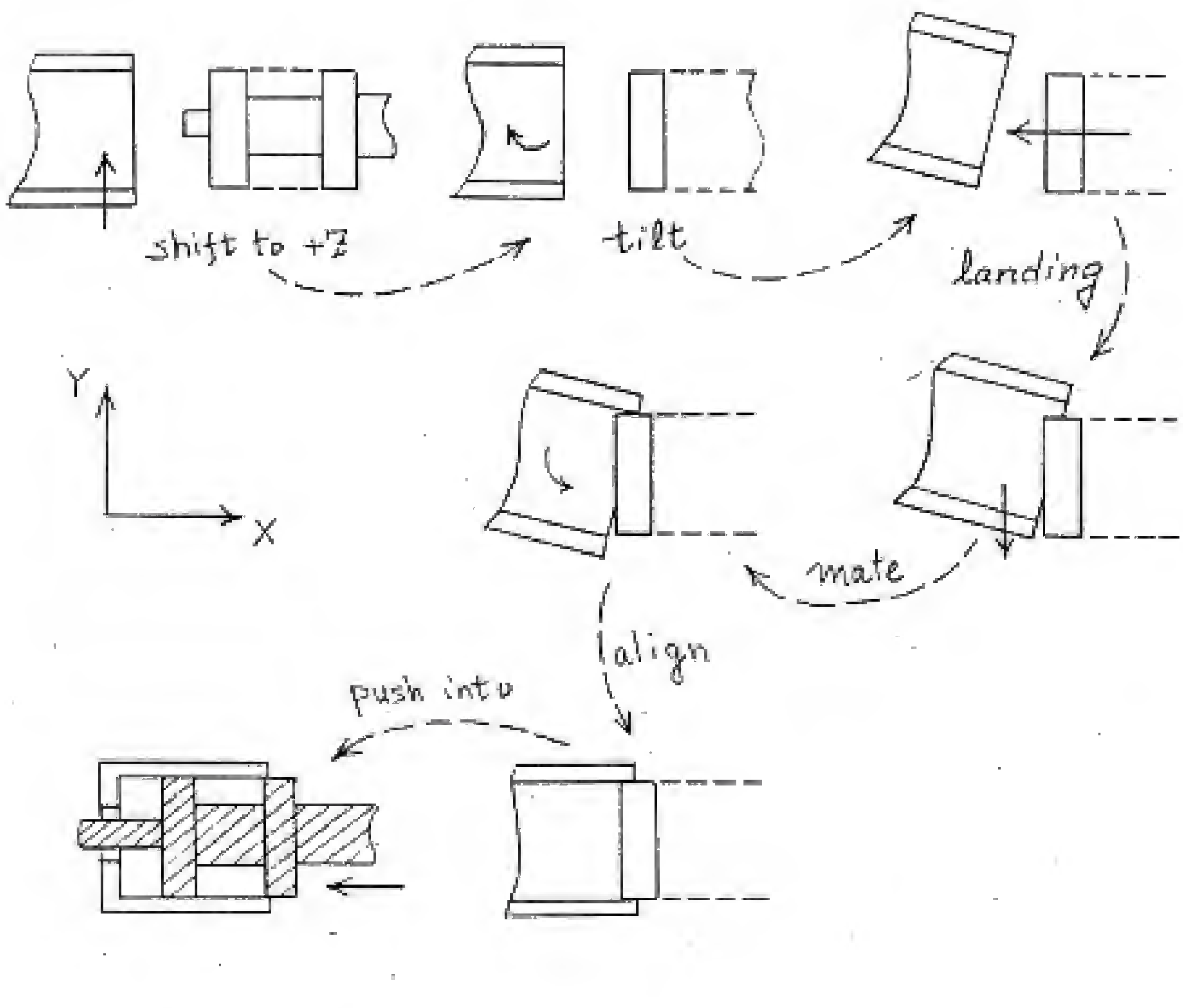


Figure 7. Peg-into-Hole assembly with "close fit".

```
(DEFUN MATE-H ( )
     (PROG (Z1)
       (FZ= small-edge-finding-force-to-"+Z") (WAIT '(?FZ))
       (SETQ Z1 (GETMF ZPOS))
       (FZ= small-edge-finding-force-to-"-Z") (WAIT '(?FZ))
       (Z= (//$ (+$ Z1 (GETMF ZPOS)) 2.0))
       (FY= small-contact-force-in-"Y-axis") (WAIT '(?FY))
       (FZ= 0)  (R= 0.0) (WAIT '(?R))
       ))
```

The next thing to do is the "push-into" action. It is the same as PUSH-INTO-Y except for the direction.

```
(DEFUN PUSH-INTO-H ( )
       (FY= 0) (FZ= 0)
       (FX= inserting-force) (WAIT '(?FX)) )
```

A stereotyped sequence of peg-into-hole assembly for close fit is defined as follows.

```
(DEFUN PEG-HOLE-C ( )
       (DROP-INTO-C) (MATE-H) (PUSH-INTO-H) )
```

3.2 Screwing a Nut.

The action for screwing a nut on a bolt is also broken into three consecutive phases; "drop-into", "mate", and "screw-into". For the "drop-into" and "mate" processes, DROP-INTO-L and MATE-Y, defined in section 3.1, are applicable. In order to screw a nut, the nut must be turned in a clockwise direction while exerting a small downward pressure. During this screwing phase, the X, Y position of the nut must accommodate to that of the screw, so X, Y force are controlled to be zero. When the robot feels the specified fastening torque, it stops the screwing motion.

```
(DEFUN SCREW-INTO ( )
       (FZ= small-down-force) (FX= 0) (FY= 0)
       (FR= fastening-torque) (WAIT '(?FR)) )
```

Thus, a procedure for screwing a nut is:

```
(DEFUN SCREW-NUT ( )
       (DROP-INTO-L) (MATE-Y) (SCREW-INTO) )
```

Again in this function, we can also ignore the function MATE-Y, because during the next function SCREW-INTO, the X, Y position of the screw adapts to the center of the nut. Thus we have an alternate, simpler function for this task.

```
(DEFUN SCREW-NUT-2 ( )
       (DROP-INTO-L) (SCREW-INTO) )
```

3.3 Picking Up a Thin Piece.

If one analyzes the assembly of any machine, one will find that the robot has to handle many small, thin pieces such as washers. It is hard to pick up such thin objects from a flat table. If the commanded position of the hand is slightly higher than the table, the robot will miss the washer, if it is a little bit lower than the table, it may exert a large force against the table. Presumably, in the latter case serious damage can occur to the robot or the table.

When picking up a thin piece, force feedback is necessary to check whether the hand makes contact with the table or not. The following is a program to pick up a thin piece on a table.

```
(DEFUN PICKUP (IT)
       (G= (+$ (DIA IT) 0.2))
       (Z= (+$ (ZOF IT) (HOF IT) 0.2) (WAIT '(AND (?Z) (?G))))
       (FZ= small-landing-force) (WAIT '(?FZ))
       (FG= grasping-force) (WAIT '(?FG)) )
```

In order to decrease the offset between the hand and the object and to allow for errors in the positioning of the part, the following

centering action is very effective.

```
(DEFUN PICKUP-2 (IT)
       (G= (+$ (DIA IT) 0.2))
       (Z= (+$ (ZOF IT) (HOF IT) 0.2)) (WAIT '(AND (?Z) (?G)))
       (FZ= small-landing-force) (WAIT '(?FZ))
       (FG= small-grasping-force) (WAIT '(?FG))
       (DG= 0.2) (WAIT '(?G))       ;open the hand 0.2 inch.
       (DR= 1.57) (WAIT '(?R))      ;turn the hand 1.57 radian.
       (FG= grasping-force) (WAIT '(?FG))  )
```

4. Outline of Assembly Demonstration

Figure 8-1 shows the initial environment of the demonstration of machine assembly. This demonstration does not use any vision program, so all information about the position of the parts must be specified. The assembly sequence is:

(1) put BEARING1 onto SHAFT

(2) put SPACER1 onto SHAFT

(3) put BEARING2 onto SHAFT

(4) assemble CYLINDER with BEARINGs in SHAFT

(5) put SPACER1 onto SHAFT, vertically

(6) put WASHER onto SHAFT, vertically

(7) put NUT on SCREW and torque down the NUT

To carry out the above demonstration, three LISP functions are defined. ASSY-L performs the peg-into-hole assembly of loose fit, as described in section 3.1.1. ASSY-C does the peg-into-hole assembly of close fit, as described in section 3.1.2. ASSY-N puts a NUT on the screw and torques it down, as described in section 3.2.

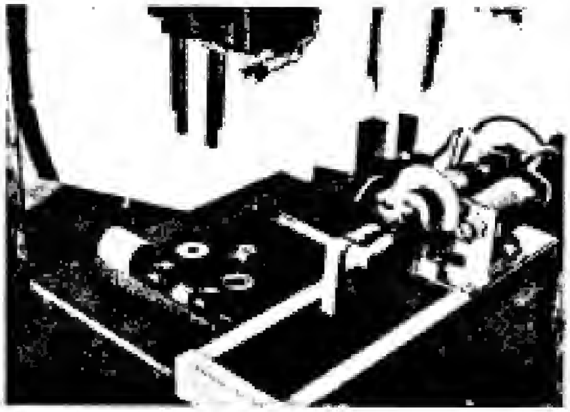
```
(DEFUN ASSY-L (PART1)
       (GO-ABOVE PART1)      ;move hand above the part
       (PICKUP-2 PART1)      ;pick it up
       (BRING-TO VSHAFT)     ;bring it to shaft
       (PEG-HOLE-L)          ;peg-into-hole of loose fit
       (RELEASE)  )          ;release


(DEFUN ASSY-C (PART1)
       (GO-ABOVE PART1)      ;move hand above the part
       (PICKUP PART1)        ;pick it up
       (BRING-TO HSHAFT)     ;bring it to shaft
       (PEG-HOLE-C)          ;peg-into hole of close fit
       (RELEASE)  )          ;release
```

```
(DEFUN ASSY-N (NUT)
       (GO-ABOVE NUT)          ;move hand above the part
       (PICKUP-Z NUT)          ;pick it up
       (BRING-TO VSHAFT)       ;bring it to the screw
       (SCREW-NUT)             ;screw a nut
       (RELEASE)  )            ;release
```

The following program ASSEMBLY carries out the demonstration completely.

Pictures in figure 8 show the experiment.

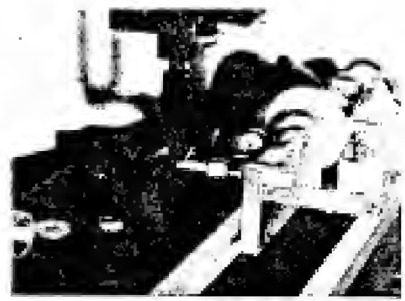
```
(DEFUN ASSEMBLY ( )
       (P= 1.57) (WAIT '(?P)) ;tilt the shaft horizontally
       (ASSY-C BEARING1)
       (ASSY-C SPACER1)
       (ASSY-C BEARING2)
       (ASSY-C CYLINDER)
       (P= 3.14) (WAIT '(?P)) ;tilt the shaft vertically
       (ASSY-L SPACER1)
       (ASSY-L WASHER)
       (ASSY-N NUT) )
```



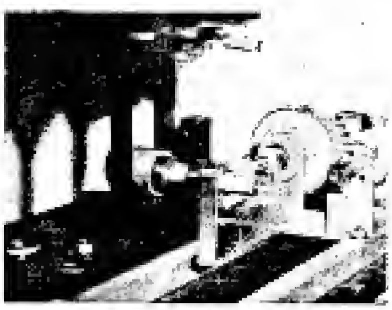
8-1. Initial environment

8-2. BEARING1

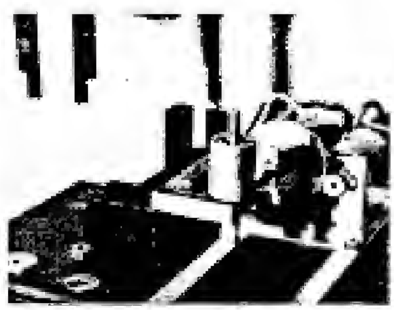
8-3. SPACER1



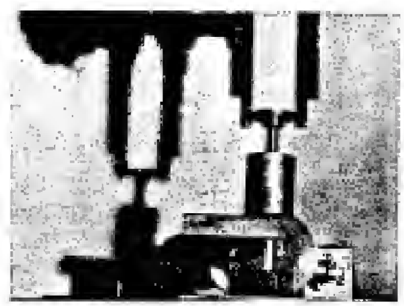
8-4. BEARING2



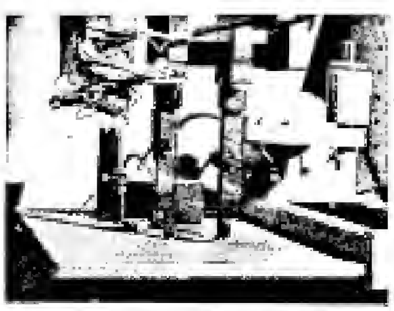
8-5. CYLINDER



8-6. Tilt shaft vertically

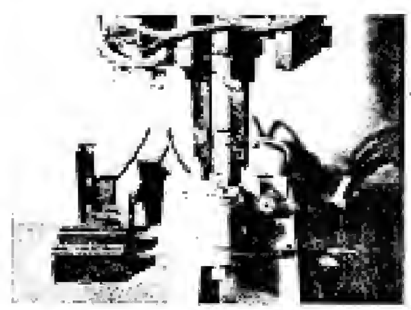


8-7. SPACER2



8-8. Pick up WASHER

8-9. WASHER



8-10. NUT

Figure 8. Assembly of the bearing complex.

5. Conclusions and Discussion.

Mating parts with close tolerance requires force feedback to assure good alignment. This paper describes a study of force feedback in a typical assembly task. The performance of the system includes such tasks as putting a peg into a hole, screwing a nut on a bolt, and picking up thin pieces such as washers. It should be noted that each of these is a basic and required task for a wide range of machine assemblies. Tolerances of 0.001 inch were achieved and experiments proved that force feedback enabled the robot to perform these assembly tasks quite reliably.

In the peg-hole assembly, the concept of loose fit and close fit were introduced. For each fit, a simple stereotyped action is presented. the stereotype for close fit uses a simple tilting technique to improve the reliability. The closest tolerance pair (CYLINDER-BEARING) assembly was assembled using this tilting technique. However, the BEARING-SHAFT and SPACER1-SHAFT assembly could be done without the tilting technique. When screwing a nut, the nut is first brought into contact with the top of screw. From that moment, force feedback guides the nut into the screw and turns it down to the prescribed torque. Force feedback also allows the robot to pick up a thin piece, even from a flat table. In this process, after the hand is felt to have arrived at the table, the robot closes its hand while keeping the contact force with the table. By using this tactic, the robot can easily pick up a thin piece whose thickness is 0.05 inch.

This study does not use any vision programs, so all the geometrical

information about the parts must be prescribed. The positional tolerance of each part is

$$\pm \left[(\text{width between two fingers}) - (\text{diameter of the part}) \right] / 2$$

When the part is located within this tolerance, the assembly can be carried out quite successfully. If the robot can see the environment, the parts need not be so carefully positioned. In a hand-eye system, the following information should be determined by a vision program.

(1) identification of the parts

(2) X, Y, Z position of the parts

(3) R-orientation of the parts

(4) maximum diameter of the parts

(5) thickness of the parts.

The force sensor complex of the Little Robot System, consisting of six L.V.D.T.s, has a maximum range of about 1 pound and a resolution of about 0.25 ounce. When the system is first started up, gravity offset calibration is done automatically. However, one cannot avoid the small hysteresis and drift that arise from the mechanical behavior of the force sensor complex. This reduces reliable measurement. In order to make the sensory system more sensitive and reliable, it seems neccessary to develop a more sophisticated force signal processing program, that compensates for the drift and the hysteresis.

Servoing is done by a machine language program running in the PDP-6, every 1/60 second. The terminating condition is checked by a time-shared LISP program in the PDP-10. Under some circumstances, this arrangement causes the robot to miss the termination condition. To prevent such timing errors, I believe that the terminating conditions

should be checked in the servoing loop on a real time basis, and that the termination should be reported to the higher level language via interrupt facilities.

Lastly, I would like to add a trivial comment: Force feedback enables the robot to guide a peg into a hole quite reliably, given that the parts do not slip in the hand. From a practical point of view, it is also important to develop a general purpose hand that prevents "slip" or that at least detects its occurrence.

Bibliography

1. Silver, D. (1973): The Little Robot System, MIT A.I. Memo 273
2. Minsky, M. (1972): Mini Robot Proposal to ARPA, MIT A.I. Memo 251
3. Inoue, H. (1971): Computer Controlled Bilateral Manipulator, Bull. of Japan Society of Mechanical Engineers,Vol.14, No.69.
4. Shirai, Y. and Inoue, H. (1973): Guiding a Robot by Visual Feedback in Assembly Tasks, Pattern Recognition, Vol.5, pp99-108.
5. Paul, R. (1972): Modelling ,Trajectory Calculation and Servoing of a Computer Controlled Arm; Stanford AIM 177.
6. Bolles, R. and Paul, R. (1973): The Use of Sensory Feedback in a Programmable Assembly System, Stanford AIM 220.
7. Goto, T. et. al. (1974): Precise Insert Operation by Tactile Controlled Robot, 2nd International Conference on Industrial Robot Technology.
8. Nevins, J.L. et. al. (1974): Exploratory Research in Industrial Modular Assembly, The Charles Stark Draper Lab. R-800.
9. Moon, D.A. (1974): MACLISP REFERENCE MANUAL, PROJECT MAC, M.I.T.